

---

*Glotzbach, Thomas; Müller, Fabian; Wenzel, Andreas; Gehr, Achim;  
Wernstedt, Jürgen :*

***Control concept for the simulation of a nonholonomic mobile  
robot with two steerable axes using MATLAB®***

---

*Zuerst erschienen in:*

Proceedings of 5th International Conference on Computational  
Intelligence, Robotics and Autonomous Systems (CIRAS 2008) :  
June 19 - 21, 2008, Linz, Austria / hrsg. von Norman Weiss ... -  
Norderstedt : Books on Demand, 2008. - ISBN 978-3-8370-6599-2

# CONTROL CONCEPT FOR THE SIMULATION OF A NONHOLONOMIC MOBILE ROBOT WITH TWO STEERABLE AXES USING MATLAB®

*Thomas Glotzbach<sup>1,2</sup>, Fabian Müller<sup>1</sup>, Andreas Wenzel<sup>1</sup>, Achim Gehr<sup>1</sup>, Jürgen Wernstedt<sup>1</sup>*

<sup>1</sup> Fraunhofer Center for Applied Systems Technology  
Am Vogelherd 50  
98693 Ilmenau, Germany

<sup>2</sup> Technische Universität Ilmenau, Institute for Automation and Systems Engineering  
P.O.Box 10 05 65  
98684 Ilmenau, Germany

## ABSTRACT

In this paper we discuss a control concept for a mobile robot. We demonstrate the robot type ASTro which is currently under development. This small- size robot has two steerable axes and will be used as a testbed system. We show the modelling for this type of robot and the realization with MATLAB®. The simulation shall enable the user to select a certain position within the mission area to command the robot to move to this point. The robot uses an event- driven, graph- based control scheme to determine whether the target is within the range it can reach directly or whether it is necessary to include an extra turn. The area borders are considered which may force the robot to perform a double bend to reach the target.

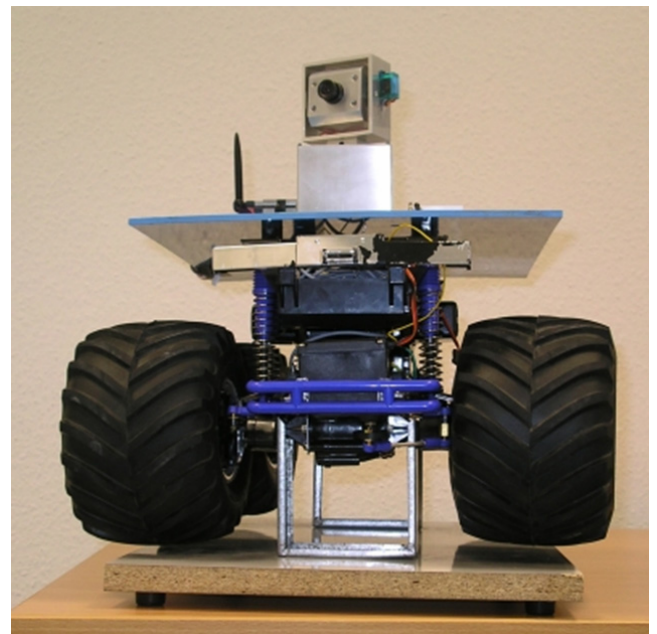
## 1. INTRODUCTION

In the framework of the development of the control software for a small autonomous land robot, software was needed to enable the robot to reach a target position given by the operator within a defined mission area. The robot is able to determine its position and has two steerable axes. A simulator was build with MATLAB® to simulate the robot executing the described task.

The nonholonomic mobile robot ASTro is a demonstrator platform which is currently under development as a joint project of the Fraunhofer Center for Applied Systems Technology and the Technische Universität Ilmenau. The chassis bases on a remote controlled vehicle of the model building which has been upgraded to allow for autonomous movement. The system has two individual motor powered, servo steered axes which will usually be steered by the same angle (only with different signs). It is controlled by a two- level hardware system with a microcontroller for the real- time interaction

with the actuators and sensors and an embedded PC for computational intensive calculations. The navigation of the robot is performed by the indoor navigation system 'Cricket' [1]. This localization system works like GPS, but uses special supersonic modules which are placed at defined positions at the ceiling and on the vehicle. Experiments at the Fraunhofer Center for Applied Systems Technology showed that within a limited mission area (5 x 5 meters) an accuracy of several centimetres can be reached. [2]

In the second chapter we will describe the modelling of the mobile robot ASTro. In the third chapter, we are going to introduce the control concept basing on the definition and calculation of several ranges around the vehicle. We will demonstrate the MATLAB® simulator in chapter five.



*Figure 1: The mobile robot 'ASTro'*

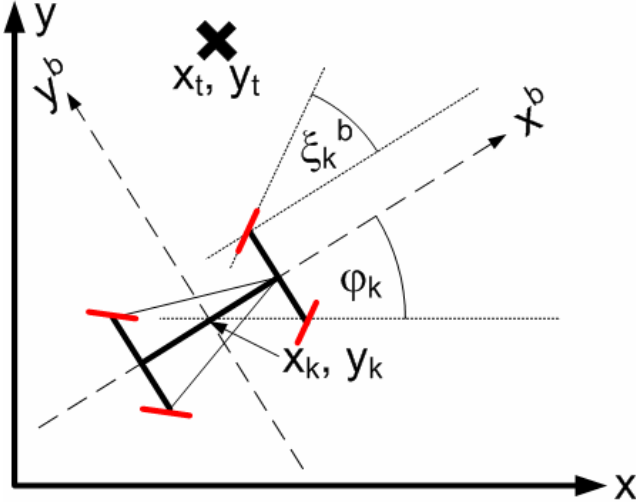


Figure 2: Definition of coordinate systems and variables

## 2. MODELLING OF THE MOBILE ROBOT

The simulation was created in MATLAB® using the GUI Layout Editor GUIDE. This application provides not only an easy way to design an appropriate GUI, but also creates the ground structure for the software as MATLAB® m- files. Both the vehicle model and the control scheme were developed within these m- files; so there was the need for a time discrete model.

For the further description we assume the following variables definition:  $(x_0, y_0, \varphi_0)$  are the position and yaw angle of the vehicle (vehicle centre point) in the mission area in global coordinates, when the new target is obtained. The position and yaw angle during the run to the target will be described with  $(x_k, y_k, \varphi_k)$  in the  $k^{\text{th}}$  time step with the step size  $T$ . The schema is shown in Figure 2. The target position, which shall be reached by the robot, is  $(x_t, y_t)$  in global coordinates and  $(x_t^b, y_t^b)$  in a vehicle body-fixed coordinate system. The steering angle is defined as  $\xi^b$ . The velocity of the kinematic centre of the vehicle is  $v_k$ , while  $v_l^b$  is the linear wheel speed.

For modelling the mobile robot ASTro a kinematic model based on the well known single-track model [3] for vehicles will be used. This model assumes that the tangential velocities of the robot's kinematics centre and of the wheels are equal to zero (no side-slip), the robot moves on a plane and that all wheels turn at the same speed. The kinematic robot model first calculates the linear velocity  $v_k$  based on the wheels linear velocity  $v_l^b$  (this velocity is adjusted by the motors) and steering angle  $\xi_k^b$ . The set values for  $v_l^b$  and  $\xi_k^b$  are the model inputs and therefore the controller outputs. The distance between the front axis and the rear axis of the robot is  $L$ . The components in x- and y-direction of the velocity of both front wheels in body fixed frame are  $v_{Fx,k}^b$  and  $v_{Fy,k}^b$ . Therefore, the following equations hold:

$$v_{Fx,k}^b = v_l^b \cdot \cos(\xi_k^b) \quad (1)$$

$$v_{Fy,k}^b = v_l^b \cdot \sin(\xi_k^b) \quad (2)$$

Also:

$$v_{Fx,k}^b = v_k \quad (3)$$

$$v_{Fy,k}^b = \frac{\varphi_k - \varphi_{k-1}}{T} \cdot \frac{L}{2} \quad (4)$$

Equation (4) is obtained by means of relative kinematics [4]. By substituting  $v_{Fx,k}^b$  and  $v_{Fy,k}^b$  of equations (1) and (2) into equations (3) and (4) and dividing (4) by (3) one retrieves:

$$\frac{\varphi_k - \varphi_{k-1}}{T} = \frac{2 \cdot v_k}{L} \cdot \tan(\xi_k^b) \quad (5)$$

By readjustment of equation (5), the yaw angle  $\varphi_k$  can be calculated (equation (9)), whereas the linear velocity  $v_k$  of the robot is obtained by evaluating equations (1) and (3):

$$v_k = v_l^b \cdot \cos(\xi_k^b) \quad (6)$$

The global position coordinates can be expressed by the coordinates of the last time step plus velocity in the particular direction times the step size  $T$ . The complete model describing the calculation of  $x_k, y_k, \varphi_k$  in relation of  $v_l^b$  and  $\xi_k^b$  is expressed by the following equations:

$$x_k = x_{k-1} + v_k \cdot T \cdot \cos(\varphi_{k-1}) = x_{k-1} + v_l^b \cdot T \cdot \cos(\varphi_{k-1}) \cdot \cos(\xi_k^b) \quad (7)$$

$$y_k = y_{k-1} + v_k \cdot T \cdot \sin(\varphi_{k-1}) = y_{k-1} + v_l^b \cdot T \cdot \sin(\varphi_{k-1}) \cdot \cos(\xi_k^b) \quad (8)$$

$$\varphi_k = \varphi_{k-1} + \frac{2 \cdot v_k \cdot T \cdot \tan(\xi_k^b)}{L} = \frac{2 \cdot v_l^b \cdot T \cdot \sin(\xi_k^b)}{L} \quad (9)$$

A white noise is added to these values to simulate typical navigation errors.

Linear wheel speed and steering angle are to be calculated by the controller in every time step. When these values are handed over to the model, time discrete PT<sub>1</sub>-elements are used to simulate a delay. If  $v_{l_{k\_set}}^b$  and  $\xi_{k\_set}^b$  are the values calculated by the controller, the following values are used as inputs for the model:

$$v_l^b = \frac{T_1}{T + T_1} \cdot v_{l_{k-1}}^b + \frac{T}{T + T_1} \cdot v_{l_{k\_set}}^b \quad \text{and} \quad (10)$$

$$\xi_k^b = \frac{T_2}{T + T_2} \cdot \xi_{k-1}^b + \frac{T}{T + T_2} \cdot \xi_{k\_set}^b, \quad (11)$$

where  $T$  is the step size of the simulation and  $T_1$  and  $T_2$  are time constants of the PT<sub>1</sub>- elements. As very strong motors and servos are used for the real vehicle,  $T_1$  and  $T_2$  can be set to a small value in comparison to  $T$ . Furthermore, it is possible to add a white noise to the final values of velocity

and steering angle to simulate inaccuracies of the actuating elements.

### 3. TARGET RANGE DETECTION

The task of the controller is to calculate velocity and steering angle for the vehicle in order to guide it to a defined target within the mission area. This area is quadratic with an edge length of five meters. In the simulator, the user shall have the possibility to click on any point within this area with the mouse. The associated position shall be handed over to the vehicle as new target.

Before the movement of the vehicle starts, the vehicle control software has to detect in which area the target is located. We defined several areas around the vehicle where different strategies will be used. It is kept in mind that the control concept shall later be implemented on a real vehicle which may not be able to perform velocity and steering angle control as accurate as in simulation. The definition of the different ranges is depicted in Figure 3. We will describe the ranges and the algorithms to test whether the defined target is within a range. First of all, four sectors are defined that equal the four quadrants of the vehicle body fixed coordinate system. Therefore, the target coordinates are transferred into the vehicle body-fixed system by rotation and translation. The right sector can be found by the sign of the coordinates according to Table 1.

$$x_t^b = (x_t - x_0) \cdot \cos(\varphi_0) + (y_t - y_0) \cdot \sin(\varphi_0) \quad (12)$$

$$y_t^b = -(x_t - x_0) \cdot \sin(\varphi_0) + (y_t - y_0) \cdot \cos(\varphi_0) \quad (13)$$

The first defined range is the Close- Up Range. If a target is within this area, it is important to perform the target approach with special diligence. The Close- Up Range is defined by a circle with radius  $dist_{CLR}$  around

Table 1: Sector Definition according to the target coordinates

	$x_t^b \geq 0$	$x_t^b < 0$
$y_t^b \leq 0$	Sector 1	Sector 4
$y_t^b > 0$	Sector 2	Sector 3

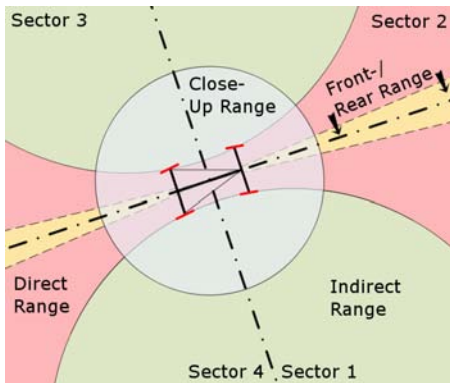


Figure 3: Defined ranges around the vehicle

the centre point of the vehicle. In the software, the distance between the target and the vehicle is calculated using the Theorem of Pythagoras and compared with  $dist_{CLR}$  to decide whether the target is within the Close- Up Range.

A Front-/ Rear Range is defined directly in front and behind of the vehicle. A target in this area can be reached without a big change of the steering angle. The area is described by an angle  $\beta$  between the area border and the  $x^b$ - axes of the body fixed coordinate system. According to Figure 3, the whole Front-/ Rear Range has a width of  $2\beta$ . For the target detection, the target coordinates in the body fixed coordinate system according to equations (12) and (13) can be used. The target is within the Front-/ Rear Range, if the following condition is fulfilled:

$$\left| \arctan\left(\frac{y_t^b}{x_t^b}\right) \right| \leq \beta \quad (14)$$

The ‘Direct Range’ is the area that can direct be reached by the vehicle in one turn. Thus it depends on the maximum steering angle  $\xi_{max}^b$  and it is checked with a small internal simulation. It may also be possible to validate this area with other methodologies, but using a simulation with the vehicle model guarantees that this approach can also be used for vehicles modelled in another way. So the simulation is based on the equations 7 – 9. At first, the target angle  $\varepsilon$  from the current vehicle position to the target is calculated using the arc tangent. Afterwards, two simulations run using positive and negative values for velocity with an absolute value of arbitrary 10% of the maximum velocity  $v_{max}$ . The two simulations are necessary because it may happen that even if the target is in one certain sector, another sector need to be cruised first to avoid collision with the border of the mission area (an example is shown below). Only 90% of the maximum steering angle is used for calculation to guarantee for a security margin. The following values were employed for the two time discrete simulations  $\lambda=1$  and  $\lambda=2$ :

$$v_k^l = a_\lambda \cdot 0.1 \cdot v_{max} \quad \text{with } a_1 = 1; a_2 = -1 \quad (15)$$

$$\xi_k^b = a_\xi \cdot 0.9 \cdot \xi_{max}^b; \quad (16)$$

$$a_\xi = \begin{cases} 1 & \text{for Sector} = (2 \vee 3) \\ -1 & \text{for Sector} = (1 \vee 4) \end{cases}$$

So in each simulation step the position values  $(x_{k1}, y_{k1})$  and  $(x_{k2}, y_{k2})$  are obtained. As soon as one of these values exceeds the area border, the particular simulation is stopped. In each step, the angles  $\delta_1$  and  $\delta_2$  from the current position of the real vehicle to the current simulated position data are calculated and compared with  $\varepsilon$ . As soon as one of the  $\delta$ - angles equals  $\varepsilon$  or is within a defined tolerance band, the simulation can be stopped. The target is within the Direct Range if the distance from the real vehicle to the target is now greater than the distance to the simulated position values. Figure 4 shows an example where at the moment ( $\delta \approx \varepsilon$ ) the target is closer to the real

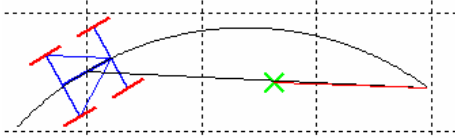


Figure 4: The target is not in the Direct Range

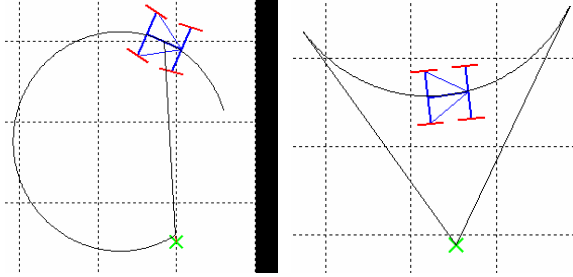


Figure 5 a & b: Targets in Direct (left) and Indirect Range (right)

vehicle as the simulated position; the target is not within the Direct Range.

If the target is in the Direct Range, the appropriate sector is stored. As said above, this can be another sector than the original one. In the situation shown in Figure 5 a, the target sector was 1, but the vehicle cannot reach the target through this sector, because it might touch the area border at the right. The vehicle needs to go through sector 4 to reach the target, so the new target sector is 4 which is important for the controller later.

If the target is not in the Direct Range, it is in the

Indirect Range, and the vehicle needs at least two turns to reach it. But it is still necessary to check whether it is possible to perform a two turn manoeuvre or whether the vehicle is too close to the area borders, and a more complex manoeuvre must be used. For this check the same simulations are employed like for the check of Direct Range. The only difference is that the sign of the steering angle according to equation (16) changes, as now the vehicle has to move in the other direction. In every step and for each simulation the positions are calculated as described above, and two angles  $\eta$  and  $\kappa$  are calculated using arc tangent:  $\eta$  is the angle between the current simulated position and the target position,  $\kappa$  is the angle between the current simulated position and the one of the last calculation step. Again, each of the two simulations is stopped as soon as the area border is touched. If this does not happen and a situation is reached where the angles  $\eta$  and  $\kappa$  are the same with a prior defined tolerance, the simulation can be stopped with success. The sector opposite of the original target sector is the 1<sup>st</sup> order evasion sector and is used if possible. The other sector next to the original target sector is defined as 2<sup>nd</sup> order evasion sector and is only employed, if the other one is not usable. An example is shown in Figure 5 b. The target is in sector 1, so the opposite sector 3 is the 1<sup>st</sup> order evasion sector and will be used, because it is eligible.

After this calculation, the position of the target is completely defined, so the control software can start to move the vehicle according to the strategies described in the next chapter.

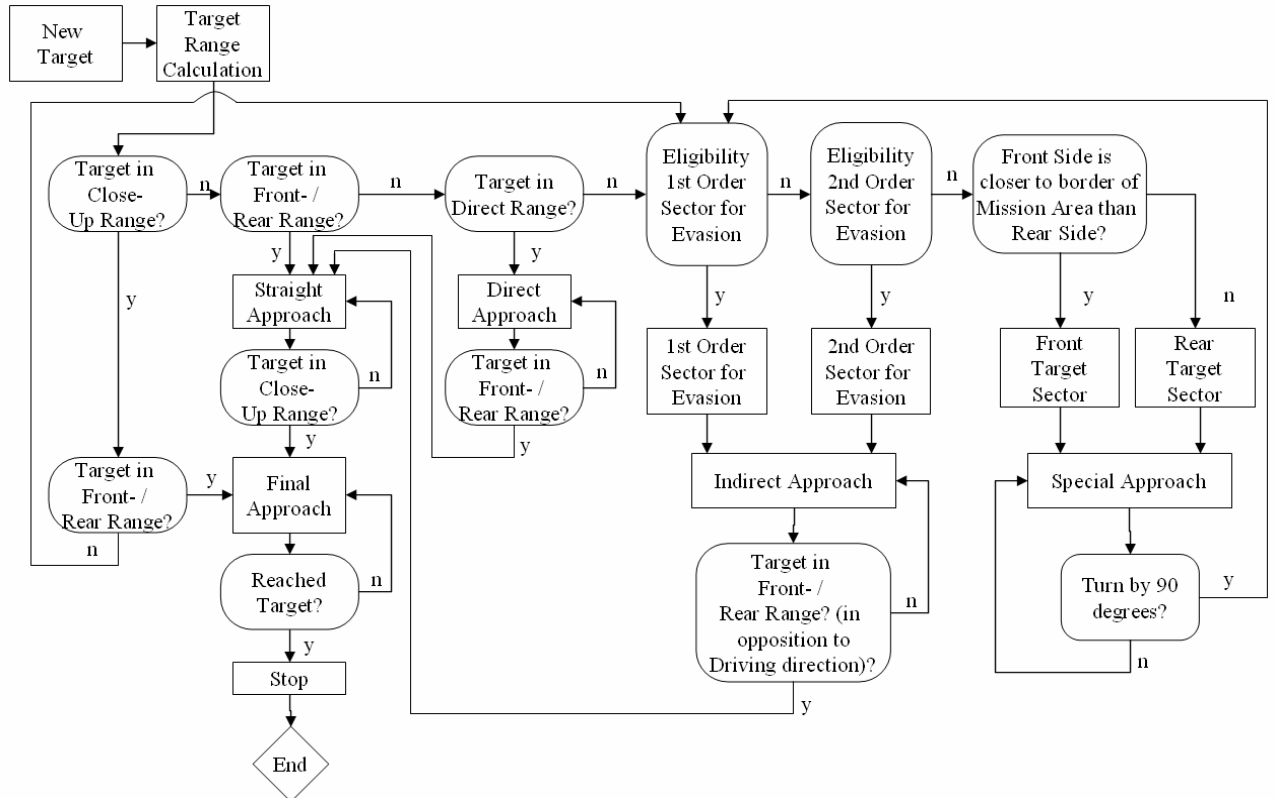


Figure 6: Flowchart of the control concept



#### 4. VEHICLE CONTROL CONCEPT

After it was detected in which ranges according to the vehicle the target is located, a control scheme is employed that is shown in Figure 6. According to the calculated target ranges, the software moves through the flowchart to determine the current valid kind of approach and to calculate the values of velocity and steering angle,  $vl_k$  and  $\xi_k^b$ , for the current time step.

If a new target was selected and the Target Range Calculation was performed, the software enters the flowchart. If the target is not in the Close- Up Range, but in the Front-/ Rear Range, the state 'Straight Approach' is entered. Here, the velocity is always set to the maximum value, with a positive sign if the target sector is 1 or 2, negative else wise. The steering angle is chosen in order to equalize the orientation angle of the vehicle to the angle  $\gamma$  between the vehicle and the target:

$$\xi_k^b = \frac{\gamma - \varphi_{k-1}}{|\gamma - \varphi_{k-1}|} \cdot \xi_{\max}^b \cdot \frac{|\gamma - \varphi_{k-1}| - 0.5 \cdot \beta}{\beta}; \quad (17)$$

$$\gamma = \arctan\left(\frac{y_t - y_{k-1}}{x_t - x_{k-1}}\right)$$

In each simulation step, the distance  $dist_t$  between vehicle and target is calculated using the Theorem of Pythagoras. If this value is smaller than the radius of the close- up range  $dist_{CLR}$ , the state 'Final Approach' is activated. In this state the calculation of the steering angle stays the same, but the velocity is reduced by multiplication with the quotient of  $dist_t$  by  $dist_{CLR}$  to guarantee for a smooth approach until the vehicle has finally reached the target.

If the target is not in the Front-/ Rear Range at the beginning, it is checked whether it is in the Direct Range, and if this is true, the state 'Direct approach' is entered. In this state, the vehicle uses an arbitrary selected value of 60% of the maximum speed and the maximum steering angle to move towards the target. The correct signs can be determined using the target sector according to Table 2. It must be kept in mind that the target sector may have been changed during the range tests (see the situation shown in Figure 5 a, for instance).

The software stays within this state until the target is in the Front-/ Rear Range. This can be determined by calculation of two angles. One is the angle between the

current position and the target, the other one is the angle between the starting position of the 'Direct Approach' and the target. As soon as these two angles are equal with a predefined tolerance, the software switches to the state 'Straight Approach' and finished the movement as it was described for this state.

According to the flowchart in Figure 6, the state 'Indirect Approach' can be reached by two ways: One possibility is that the target is neither in the Close- Up Range, nor in the Front-/ Rear Range, nor in the Direct Range. The other possibility is that the target is in the Close- Up range, but not in the Front-/ Rear Range. In this case, always the 'Indirect Approach' is used, even if the target is in the Direct Range. This is done because the target may be very close to the vehicle. Even if a 'Direct Approach' may be possible, there may be difficulties in the reality because of inaccuracies of the actuators. To be on the safe side, an 'Indirect Approach' is used that will at first enlarge the distance to the target and then allow an ease cruise to the target with the 'Straight Approach'.

If an 'Indirect Approach' is necessary and one of the two evasion sectors is useable, the software enters the adequate state and moves the vehicle in the direction of the chosen sector, using the actuator values according to Table 2. This state stays active, until the target is in the Front-/ Rear Range which can be checked by calculating and comparing the two angles  $\eta$  and  $\kappa$ , according the description given at the end of chapter 3. After these angles are equal, the first turn is over. The vehicle needs to stop and change the direction. A new Target Sector Calculation is performed; afterwards the software can shift to the state 'Straight Approach'.

In several situations, the vehicle may be very close to the area borders. Both evasion sectors may not be usable, like it is shown in Figure 7 a. A 'Special Approach' with a double bend is necessary. At fist, it is decided whether front or back side of the vehicle is closer to the area border. Then the direction is chosen where there is more space. The vehicle performs a 'Direct Approach', as it was described before, although the target is not reachable this way. After the orientation of the vehicle,  $\varphi_k$ , has changed by 90 degrees (or -90 degrees, according to the

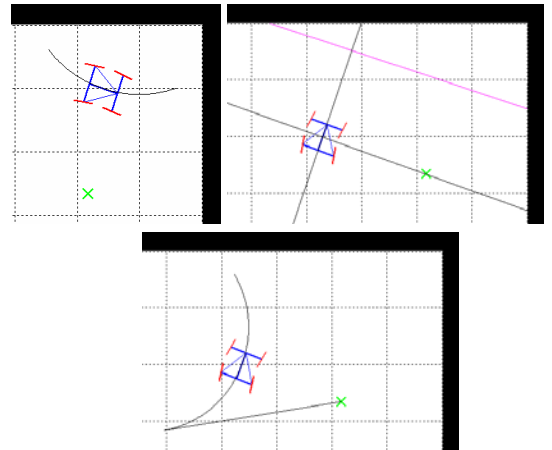


Figure 7 a, b and c: Usage of a double bend

Table 2: Actuator Values in 'Direct Approach'

	$vl_{k\_set}$	$\xi_{k\_set}^b$
Sector 1	$0.6 \cdot vl_{\max}$	$-\xi_{\max}^b$
Sector 2	$0.6 \cdot vl_{\max}$	$\xi_{\max}^b$
Sector 3	$-0.6 \cdot vl_{\max}$	$\xi_{\max}^b$
Sector 4	$-0.6 \cdot vl_{\max}$	$-\xi_{\max}^b$

chosen direction), the direct approach is stopped (see Figure 7 b); the first part of the double bend was performed. Now the software performs again a Target Sector Calculation and goes back to the calculation of the evasion sectors. Afterwards, a regular 'Indirect Approach' is performed. (see Figure 7 c). As the vehicle has moved away from the area border, at least one evasion sector will be available now. The turn of the indirect approach is the second part of the double bend. After this manoeuvre, the software can switch to the state 'Straight Approach' and finish the mission.

With the described algorithm, it is possible to reach every given target within the mission area. The flowchart covers all possible situations and provides strategies to finish the mission successfully.

## 5. SIMULATION WITH MATLAB®

A simulator was developed with MATLAB® to show both the Target Range Calculation and the movement of the vehicle to the target. The dialog of this program is shown in Figure 8. It is possible to enter the start position and orientation and point at the target. After this, the program is started and performs the different range tests, as explained before. The proceedings of the tests are demonstrated graphically. At the moment shown in Figure 8, the Front-/ Rear Range test is performed. When all range tests have been executed, the software starts to move the vehicle to the target, using the explained algorithms. Important data, like position, orientation, steering angle and velocity of the vehicle as well as the time are shown in the lower right part of the dialogue. The user is also able to change global parameters, like maximum velocity and steering angle, simulation step size, all described range parameters and details of the white noise which is used to simulate the sensor and actuator inaccuracies. So this software can be used to determine which abilities the real ASTro vehicle must have to fulfil several target approaching tasks.

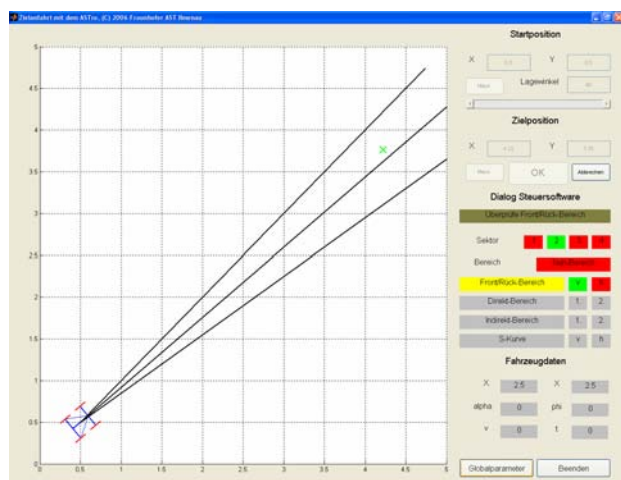


Figure 8: The dialog of the MATLAB Simulation

## 6. CONCLUSION

We presented an algorithm to realize a controller for a nonholonomic mobile robot with two steerable axes. The algorithm bases on the definition of several ranges around the vehicle. The target ranges are detected, and the concrete actuator values for the robot are calculated according to the ranges. The software for the vehicle model and the controller was realized and demonstrated in a MATLAB® simulation. Several real constraints, like actuator and sensor inaccuracies have been considered. This will allow the employment of the control software for the real ASTro vehicle.

## 7. REFERENCES

- [1] M. I. T. Computer Science and Artificial Intelligence Laboratory, Homepage of the Cricket localization system; <http://cricket.csail.mit.edu/>; visited on 2008-04-16
- [2] Guß, C., "Development of algorithms for a global indoor localization system with supersonic base (Entwicklung von Algorithmen für ein globales Indoor-Lokalisationssystem auf Ultraschall-Basis)". Seminar Paper of Technische Universität Ilmenau, 2006
- [3] Muir, P.F.; Neumann, C.P., "Kinematic Modeling of Wheeled Mobile Robots", *Journal of Robotic Systems*. Vol. 4, no. 2, 1987
- [4] Hibbeler, R.C., *Engineering mechanics: dynamics*, Singapore [u.a.] : Prentice Hall, 1997